

Synchronization for IP-Layer TDM on Ad-Hoc Wireless Mesh Networks

Kevin Duff, Peter Clayton, and Alfredo Terzoli
Rhodes University, Grahamstown, South Africa

Abstract—IP-Layer TDM requires accurate synchronization. This document describes a synchronization scheme which is suitable for ad-hoc networks. The scheme leads to loose synchronization of the clocks of nodes with those of their neighboring nodes, and does not depend on any central or real clock. It operates in a completely distributed manner.

Index Terms—Ad-Hoc Network, Distributed TDM, Synchronization,

I. INTRODUCTION

Time division multiplexing (TDM) is normally implemented at the MAC layer on the OSI stack. Two recent initiatives [3,4] have shown that TDM can also be done at the IP layer in infrastructure 802.11 networks.

TDM-based MAC schemes based on slotting algorithms [1,6,8] have been proposed for ad-hoc wireless mesh networks.

Our research project investigates the feasibility of ad-hoc TDM at the network layer. The project borrows concepts from slotted ad-hoc MAC schemes, but implements the TDM functionality at Layer 3.

A. Ad-Hoc Wireless Mesh Networks

An ad-hoc wireless network consists of homogeneous nodes which can act as routers and thus forward data for other nodes. There exists no central infrastructure in an ad-hoc network, thus contention for the medium must be resolved in an entirely distributed manner.

A mesh network has multiple routes between nodes, which increase the network's resilience against failure. Ad-hoc networks commonly have a mesh topology.

In an ad-hoc wireless mesh network, the addition of a new node should theoretically extend the coverage area of the network, provide redundant routes and thereby improve the bandwidth capacity of the network. Such networks thus offer great promise.

B. The Hidden Node Problem

The hidden node Problem in 802.11 DCF networks is a

result of the CSMA/CA MAC scheme used. Under CSMA/CA, a node listens to ensure that no other transmissions are in progress, before transmitting. The problem occurs when two nodes, which are out of receive range of one another, attempt to transmit simultaneously to a common node. The receiving node experiences a collision, and the data is lost.

Ad-hoc wireless mesh networks are multi-hop networks, in that data is forwarded by intermediate nodes in order to extend the coverage area of the network. Such networks, by definition, will have a large number of hidden nodes. Hidden node problems can severely degrade the performance of a mesh network.

Various solutions to the hidden node problem in ad-hoc wireless mesh networks have been proposed. The most promising solutions appear to be those based on TDM or slotting. Some of these solutions are briefly discussed under the Related Work heading.

C. Ad-Hoc TDM at the network layer

The research presented in this paper is part of a larger project to investigate ad-hoc TDM at the network layer of the OSI reference stack. The project aims to combine techniques from various sources, discussed in the next section, in a novel way.

The project hopes to demonstrate that the hidden node problem can be eliminated in 802.11-based mesh networks, with existing hardware, working only with Layer-3 and above on the OSI stack. Although designed with 802.11 networks in mind, the scheme is also applicable to other wireless network types.

If the project succeeds, it could dramatically improve the performance of community ad-hoc wireless mesh networks, based on low-cost single-transmitter 802.11 equipment,

The proposed scheme is not well suited to low-power sensor networks, because nodes need to listen continuously and thus cannot "sleep".

The scheme is also not intended for mobile ad-hoc networks, because it assumes that the network will be relatively consistent from moment to moment, and does not adapt well to frequent topology changes.

This paper focuses on the challenge of synchronizing a node's clock to the clocks of nearby nodes.

A novel aspect of the synchronization scheme is that it does not assume symmetrical delays between nodes, and can adapt to networks in which delays are asymmetrical.

Manuscript received May 1, 2006.

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, Tellabs and StorTech THRIP, and the National Research Foundation.

II. RELATED WORK

A. IP-Layer TDM

Frottle [1] schedules traffic from each client in an infrastructure network, with a master-node passing tokens to clients to co-ordinate. Frottle is a userspace Linux application, which uses the iptables QUEUE functionality via the libipq library. Frottle was created to overcome the crippling effects of the hidden node problem in large-scale outdoor community networks.

The developers of Frottle have initiated a project [3] to develop a version of Frottle suitable for ad-hoc mesh networks, but this project is a long way from completion.

WiCCP [4] is a project with similar aims to Frottle, although it is implemented in the Linux kernel rather than userspace. Like Frottle, it employs a token-passing scheme in a network with a master and slaves.

These projects demonstrate that IP-Layer TDM is feasible in infrastructure networks.

B. Slotted Ad-Hoc MAC Schemes

ADHOC-MAC [1] implements a dynamic TDMA mechanism that is able to provide variable-bandwidth and reliable channels needed for QoS delivery. Dynamic TDMA is achieved by the Reliable R-ALOHA protocol (RR-ALOHA) [2], a distributed reservation protocol capable of dynamically establishing a reliable single-hop broadcast channel, which provides knowledge of MAC transmissions in overlapping segments. Unlike R-ALOHA, RR-ALOHA requires no central repeater and operates in a completely distributed manner.

SYN_MAC [8] is another slotted MAC scheme. It claims to outperform ADHOC-MAC, but requires network-wide synchronization. Nodes reach system-wide synchronization by detecting the pilot signals in cellular systems and/or the GPS signals from satellites

FPRP [6] is similar to ADHOC-MAC. The protocol performs the tasks of channel access and node broadcast scheduling, allowing nodes to make reservations within TDMA broadcast schedules. It employs a contention-based mechanism with which nodes compete with each other to acquire the TDMA slots.

C. Synchronization

The NTP and SNTP protocols [9] are able to accurately synchronize the clocks of nodes to a common clock. NTP estimates delay time to be half of "the total delay minus remote processing time", by timing the return-trip of a packet and assuming symmetrical delays. Once the delay time is known, the protocol is able to accurately synchronize one clock to another.

However, NTP requires some reference clock that defines the true time in order to operate. The need for a central reference clock does not suit the ad-hoc paradigm.

In [7], an interesting MAC scheme for ad-hoc sensor networks is discussed. The scheme requires periodic synchronization among neighboring nodes to overcome the effects of clock drift rates, leading to a loose synchronization between neighboring nodes. Network-wide synchronization is not required, and indeed they point out

that two neighboring nodes may have entirely different schedules if they each in turn synchronize with different nodes. The scheme does not account for latencies in beacon transmissions.

III. AD-HOC TDM AT THE NETWORK LAYER

Our project borrows concepts from slotted ad-hoc MAC schemes, but implements the TDM functionality at Layer 3.

An algorithm based on RR-ALOHA is used by each node to reserve a repeating time *slot* within a repeating *frame* of n slots. Once a node has reserved a slot, only that node is allowed to transmit for the duration of the slot. The node is not allowed to transmit at other times.

We use the iptables QUEUE functionality in libipq to schedule transmissions, in a similar manner to Frottle. This functionality allows us to transmit a predetermined amount of data during each slot.

Synchronization of nodes within the network is essential for RR-ALOHA to operate correctly.

IV. SYNCHRONIZATION

In order to be able to perform time division multiplexing, a node needs to be able to accurately synchronize its clock with the clocks of neighboring nodes. There is no need for synchronization with any "real" or central time.

Synchronization is achieved through beacons, which are sent as UDP broadcasts by each active node at the beginning of that node's slot. Each node synchronizes to the last beacon which it received. Refer to Table I for a description of the main fields which a beacon contains.

This scheme leads to a loose synchronization, under which the clocks of nodes are synchronized closely to those of nodes within transmit range, but are not necessarily synchronized to nodes which are out of range.

Because some nodes are able to transmit with a lower latency than others, the beacon arrival times will be irregular. The receiving node is able to compensate for this if it knows the latency associated with the transmission.

TABLE I
BEACON PACKET: FIELD DEFINITIONS

Field	Meaning
<i>SI</i>	Slot Index
SenderID	The unique ID (address) of the sending node
Latency	The sender's estimate of own latency, includes i -value
Slot_table	RR-ALOHA slot allocation table
SyncNode	Node to which this node is synchronized
e_x	Estimate of latency which this node used when synchronizing, including i_x

A. Assumptions

We assume that the latency associated with a node remains constant. The effects of signal propagation delays, which would lead to far-off receivers experiencing greater latencies, are not considered in this document. We assume that the latencies associated with lower-layer traffic overheads remain constant.

We do not make allowances for processing latencies on nodes. Latency measurements include any processing

latencies which occur while taking the measurement, but it is assumed that the variance of such latencies is negligible in our context.

We assume a good signal-to-noise ratio, so that retransmissions due to errors are rare.

We assume that all links between nodes are bidirectional, (half duplex, because the channel is shared and only one node may transmit at a time).

B. Latency

For the purposes of this paper, we shall define the latency of a node X as the time difference between the moment when node X transmits a beacon, and the moment when other nodes receive that beacon.

Because beacons are broadcast, it is fair to assume that receiving nodes will receive the broadcast simultaneously. As previously stated, signal propagation delays are not considered.

The latency of a node in an 802.11 network is of the order of milliseconds, which is sufficient to skew any time-based calculations which nodes might perform. Thus it is important for nodes to compensate for latency.

We have determined experimentally on an 802.11b network in the laboratory that the latency associated with a node's beacon transmission on an otherwise idle network remains fairly constant, with a variance of approximately 3ms. This indicates that node synchronization to within 3ms is feasible. By optimizing network parameters, it might be possible to reduce this variance.

In order to synchronize, a node needs to know the latency of the node to which it is attempting to synchronize. The receiving node adds this latency to the arrival time of the beacon, and uses the sum as a timestamp for synchronization.

C. Non-contiguous Slots

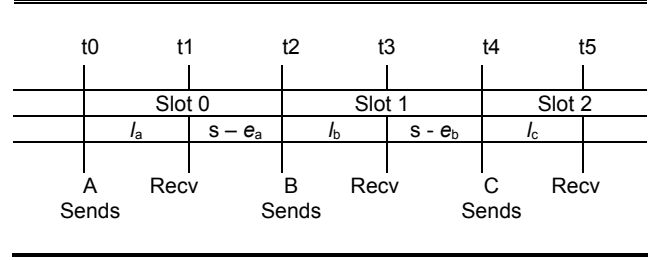
For the sake of simplicity, the calculations which follow assume that beacons are received in contiguous slots. We assume that node A reserves the first slot, while B and C reserve the second and third slots, respectively.

The calculations are equally applicable across multiple slots, if $(n*s)$ is substituted for s in calculations, such that n is a positive number representing difference between the slot numbers of nodes A and B.

V. DETERMINING LATENCY

Depending on the configuration of nodes, various methods are proposed to determine the latency of nodes in a network. Refer to the timeline in Figure I when reading the calculations which follow.

FIGURE I
GRAPHICAL TIMELINE



The table which follows introduces the variables which are used in the latency calculations which follow:

TABLE II
DEFINITIONS OF IMPORTANT VARIABLES

Variable	Meaning
l_x	The latency of Node x
t_x	Transmit time of beacon by Node x
r_x	Arrival time of beacon received from Node x
e_x	Estimate of l_x , by a node synchronizing to Node x
i_x	Indicator of accuracy of e_x , see table III.
s	Slot duration

A. Two Nodes: Sum of Latencies

If node B has previously synchronized to node A, the sum of their latencies can be determined by B at t_3 as follows:

$$r_b = l_a + (s - e_a) + l_b \quad (1)$$

which can be rewritten as:

$$l_a + l_b = r_b + e_a - s \quad (2)$$

Node A is able to measure r_b , the arrival time of the beacon from B. Node B broadcasts e_a (the estimate of A's latency which node B used in its calculations) with its beacon, thus node A learns e_a .

Should either of l_a or l_b be known, it is trivial to calculate the other and then achieve synchronization by rewriting (2):

$$l_a = r_b + e_b - s - l_b \quad (3)$$

B. Two Nodes: Half Return Time Method

Although two nodes can easily calculate the sum of their latencies, they cannot easily determine their respective individual latencies.

In a homogeneous network, it is fair to assume that the send latencies of two nodes are equal.

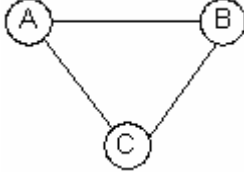
Thus, like NTP, we assume symmetrical latencies and halve the sum of the latencies:

$$l_a = l_b = (r_b + e_b - s) / 2 \quad (4)$$

C. Three Interconnected Nodes: Measuring Latency

Where 3 or more interconnected nodes exist on a network, latency may be measured by the 3rd node as follows:

FIGURE I
A CLUSTER OF 3 INTERCONNECTED NODES



Node C listens for the beacons from two nodes, A and B, such that B is synchronized to A's beacon. At t_3 , when B's beacon is received, node C can calculate l_b according to this formula:

$$l_b = r_b - r_a - (s - e_a) \quad (5)$$

Having learned B's latency, node C synchronizes to B's beacon in the usual manner:

$$t = r_b + l_b \quad (6)$$

Node C waits for the start of its slot at t_4 , then transmits a beacon stating that it is synchronized to B, stating B's latency, so that node B learns its latency from C.

Node A can now measure Node C's latency in the same way at t_5 :

$$l_c = r_c - r_b - s - e_b \quad (7)$$

and synchronize to node C at time t_5 . Node A's latency is measured in the same way.

VI. PROPAGATION OF LATENCY INFORMATION

It is desirable for nodes to learn of their own latencies, so that they can broadcast this information in their beacons. This enables any nodes which receive the beacon to accurately synchronize their clocks to the node.

A. Partially Connected Nodes

There are various ways that latency measurements can propagate through the network.

Any node A which is connected to any other node B of known latency l_b , can calculate its latency according to equation (3):

$$l_a = r_b + e_b - s - l_b$$

but l_b is known, so $e_b = l_b$ thus

$$l_a = r_b - s \quad (8)$$

If there exists anywhere in the network an interconnected cluster of 3 nodes, those nodes can measure their respective latencies. Using equation (8), the latencies of other nodes can then be determined.

Any node which is aware of its own latency will broadcast this latency in its beacon. Thus any node which receives the beacon of that node can accurately synchronize to the sender.

A node which receives a beacon may not be able to determine its own latency unless the sender chooses to inform the receiver of this. Should this be the case, the

receiver can issue a Calibration Sync request to determine its latency.

B. Calibration Sync Requests

A node which has been unable to determine its own send latency, because no other node has informed it of this, can issue a Calibration Sync request. As a prerequisite, the node must be synchronized to another node. The Calibration Sync request instructs all nodes which receive the request, and are able to measure the node's latency, to synchronize to the requesting node. The requesting node's latency estimates are reported by each of the receivers during their beacon broadcasts.

C. Comparing the Accuracy of Estimates

Nodes are informed of their latencies through estimates which are sent by other nodes. This document has described various methods by which latency estimates might be derived.

Some methods yield better estimates of latency than others. A node which has measured the latency of another using method C of Determining Latency, will have a more accurate estimate than a node which has used method B and halved the return time.

We thus need a way of indicating the accuracy of a latency estimate. This would enable nodes, offered various estimates by other nodes, to select the most accurate estimate of latency for use in their calculations.

For this purpose, we introduce a value i_x , known as the i-value, which indicates the accuracy of the corresponding estimate e_x .

Table III enumerates the meanings of values for i_x .

When a node uses method A of Determining Latency, it knows the sum of two latencies, $l_a + l_b$. If it learns l_b from the beacon of another node which has measured l_b , the node can calculate l_a . Such a calculation yields a second generation, or 2G measurement. Should another node use this estimate of l_a , a 3G measurement results.

Thus if a node uses method A and a received estimate of latency in a calculation, the received i-value should be incremented by the receiver to indicate an assumed loss of accuracy.

TABLE III
I-VALUE MEANINGS

i-value	Meaning
0	Measurement / Reported Measurement
1	2G Measurement (measurement based on an estimate)
2	3G Measurement
3..9	4G..10G Measurement
10	Precalibrated (determined by hardware, software and speed combo)
20	Half Return Time (HRT) Estimate
21..29	2G.. 10G HRT Estimate
100	Initial Guess

VII. NETWORK FRAGMENTATION AND ASSOCIATION

Ad-hoc networks may become fragmented, when a group

of nodes become disconnected from the main network. This might happen if a node, which carries data between a group of other nodes and the main network, fails. Two independent networks result.

Ad-hoc networks can conversely associate, when connectivity is established between two previously independent networks.

The Slot Index, described below, provides a means of resynchronizing the clocks of a large group of nodes when networks associate.

A. Slot Index

The Slot Index (SI) is an integer representing the number of slots which have elapsed since the network started. It indicates the age of a network, and each node broadcasts the SI within every beacon packet. Each node has the ability to calculate the SI on-the-fly as follows:

$$SI = b + n \quad (10)$$

where b is the value of the SI reported by the last node to broadcast a beacon and n is the number of complete slots which have elapsed since that time, as measured by the node's internal clock.

B. A Young Network Associates with an Older One

The SI can be used to synchronize two entire networks when an older and a younger network merge.

When a newer network connects with an older network, the SI can be used to grant precedence to the older (or the newer) network's clock, enabling the nodes to resolve which clock is adopted. This can be achieved if nodes enforce the rules described below each time they receive a beacon:

- ◇ *If the received SI is greater than my SI, let my SI equal the received SI and synchronize my clock to the received beacon.*
- ◇ *If the received SI is less than my SI, ignore the beacon transmission.*

The rules will ensure that the clock of the older network is adopted by all nodes in the network.

C. A Fragmented Network Reassociates

A more difficult problem is faced when an existing network is fragmented into two networks, then reassociates. Over time, the two networks will lose synchronization. When the two networks reestablish connectivity, perhaps after the repair of a faulty node which was connecting the two, their Slot Indices might be identical. However, their clocks are might be out of synchronization by a fraction of a slot.

In this case, the connecting node (X) will be unable to synchronize properly to either network.

FIGURE III
EXAMPLE OF A CONNECTING NODE (X)



Such a connecting node X can detect its status as follows:

*If node A is synchronized to node B
and I am synchronized to node C
and I cannot hear node B
and node A remains out of synchronization with me after
my beacon broadcast
then I am a connecting node.*

To resolve the problem, the connecting node (X) must increment the Slot Index. This forces all the nodes which receive the incremented SI to adopt the "older", incremented SI generated by the connecting node.

However, if two connecting nodes on opposite ends of a large network employ this tactic, and increment their respective SIs by one, there will still be two unsynchronized clocks present in the network and nodes will still be unable to decide which one to adopt.

This situation can be remedied if a node increments the SI by a random positive integer, rather than just by 1. Should two nodes simultaneously decide to increment the SI, the highest of the two new SIs will be adopted and the other discarded.

This synchronization scheme will lead to ripples of clock instability through the network each time a connecting node increments the SI. For this reason, the scheme is better suited to fixed rather than mobile networks.

VIII. FUTURE WORK

Work is underway to implement, test and refine the algorithms presented in this paper.

It is unknown how well the algorithms will perform in an outdoor environment, across large distances, or when signals degrade resulting in retransmissions.

The behavior of the proposed algorithms needs to be studied in large networks, via simulation, to ensure that they lead to convergence and a stable clock.

While preliminary tests in the laboratory on a small wireless network have been successful, the behavior of larger numbers of real nodes has not been studied.

Little attention has been paid in this paper to the payload data, which is transmitted after the beacon during a node's slot under the TDM scheme. The libipq mechanism enables us to transmit a predetermined amount of data during each slot, but we require a means of calculating an optimum amount of data to transmit.

If a node transmits too much data, it will over-run its slot and interfere with the slots of other nodes. If the node transmits too little data, bandwidth is wasted. The amount of data which can be transmitted is dependent on variables such as transmission speed, latency, signal-to-noise ratio, etc.

IX. CONCLUSION

We proposed a scheme to achieve a loose synchronization of clocks within an ad-hoc network. The scheme operates in a completely distributed manner.

The synchronization scheme is part of a larger scheme to effect TDM at the network layer.

Because the scheme is implemented at Layer 3, implementation requires no modification to underlying network stacks, device drivers or firmware. It can be implemented on existing network hardware.

If the scheme proves viable, it has the potential to eliminate the effects of the hidden node problem in ad-hoc wireless mesh networks. Without the hidden node problem, single-transmitter ad-hoc meshes could offer significantly better performance than they do today.

REFERENCES

- [1] F. Borgonovo, A. Capone, M. Cesana, Luigi Fratta, "ADHOC MAC: a new MAC architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services," *ACM Wireless Networks (WINET)*, July 2004, Volume 10, Issue 4, Page(s): 359-366 [Available: <http://www.elet.polimi.it/upload/antlab/RESEARCH/Ad-hoc/papers/WINET03.pdf>] [Accessed: 23 Apr 06]
- [2] F. Borgonovo, A. Capone, M. Cesana, L. Fratta, *RR-ALOHA, a Reliable R-ALOHA broadcast channel for ad-hoc inter-vehicle communication networks*, in proceedings of the 1st Mediterranean Ad Hoc Networking Conference (Med-Hoc-Net), Chia (CA), Italy, September 4-6 2002. [Available: <http://www.elet.polimi.it/upload/antlab/RESEARCH/Ad-hoc/papers/medhocnet2002.pdf>] [Accessed: 23 Apr 06]
- [3] Frottle Webpage, <http://frottle.sourceforge.net>, [accessed 25 Mar 06]
- [4] WiCCP, <http://patraswireless.net/software.html>, [accessed 25 Mar 06]
- [5] MWRPAdhocFrottleDev, <http://www.melbourne.wireless.org.au/wiki/?MWRPAdhocFrottleDev>, [accessed 25 Mar 06]
- [6] Chenxi Zhu, M. S. Corson, *A Five-Phase Reservation Protocol (FPRP) for Mobile Ad Hoc Networks*, *Wireless Networks*, Volume 7, Issue 4, September 2001.
- [7] Wei Ye, John Heidemann, and Deborah Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of IEEE Conference on Computer Communication (INFOCOM'02)*, 2002, pp. 1567-1576.
- [8] Hongyi Wu, Anant Utgikar, and Nian-Feng Tzeng, "SYN-MAC: A Distributed Medium Access Control Protocol for Synchronized Wireless Networks", <http://www.cacs.louisiana.edu/~wu/paper/SYNNMAC.pdf>
- [9] The NTP FAQ, NTP Homepage, <http://www.ntp.org> [accessed: 25 Apr. 06]

Kevin Duff is reading toward his masters degree in Computer Science at Rhodes University. His current area of interest is ad-hoc wireless mesh networking.